ITESO, UNIVERSIDAD JESUITA DE GUADALAJARA

UART Specification

Embedded Systems

Francisco Martinez Chavez

9-Oct-17

ITESO

Universidad Jesuita
de Guadalajara

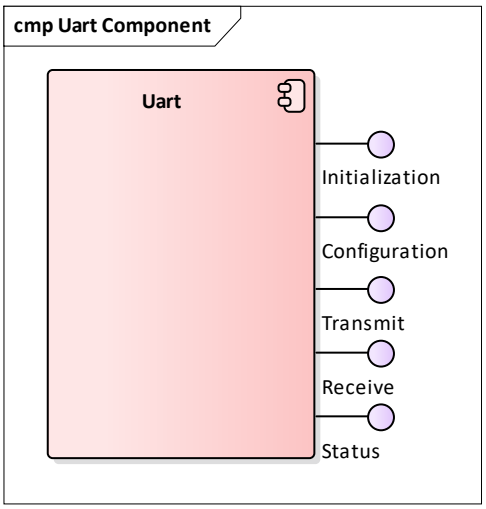
Author Francisco Martínez Chávez

Date 9-Oct-17

1. TABLE OF CONTENTS

2. FUNCTIONAL SPECIFICATION

This document specifies the basic functionality, API and the configuration of the UART driver module. The UART driver supports initialization, transmit and receive interfaces to meet asynchronous serial communication.



1 UART COMPONENT DIAGRAM

The Uart component provides static initialization and runtime configuration interfaces as required by the application. In addition supports transmit and receive operation functions.

3. UART API SPECIFICATION

The specified interfaces and types defined next have to be fulfilled by the UART driver.

3.1. IMPORTED TYPES

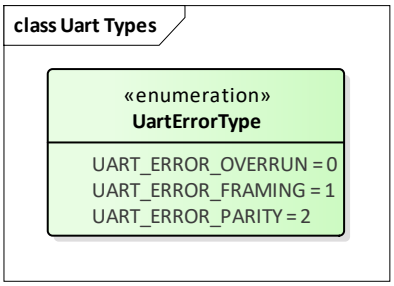
Here, all the other types imported from other modules are listed.

Module	*Imported Type*
Std_Types	Standard types, formerly named typedefs.h

3.2. TYPE DEFINITIONS

3.2.1. UART ERROR TYPE

The following type definitions shall be exported from *Uart_Types.h* file.

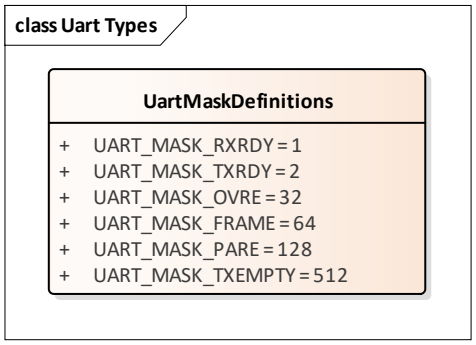


These errors provide the occurred error to upper layers through callback notifications. Refer to Function Definitions chapter for more information of the usage.

Name	*UartError*
Type	enumeration UartErrorType
Description	Uart Error
Range	0..2

3.2.2. UART MASK DEFINITIONS

The following definitions shall be exported from *Uart.h* file.



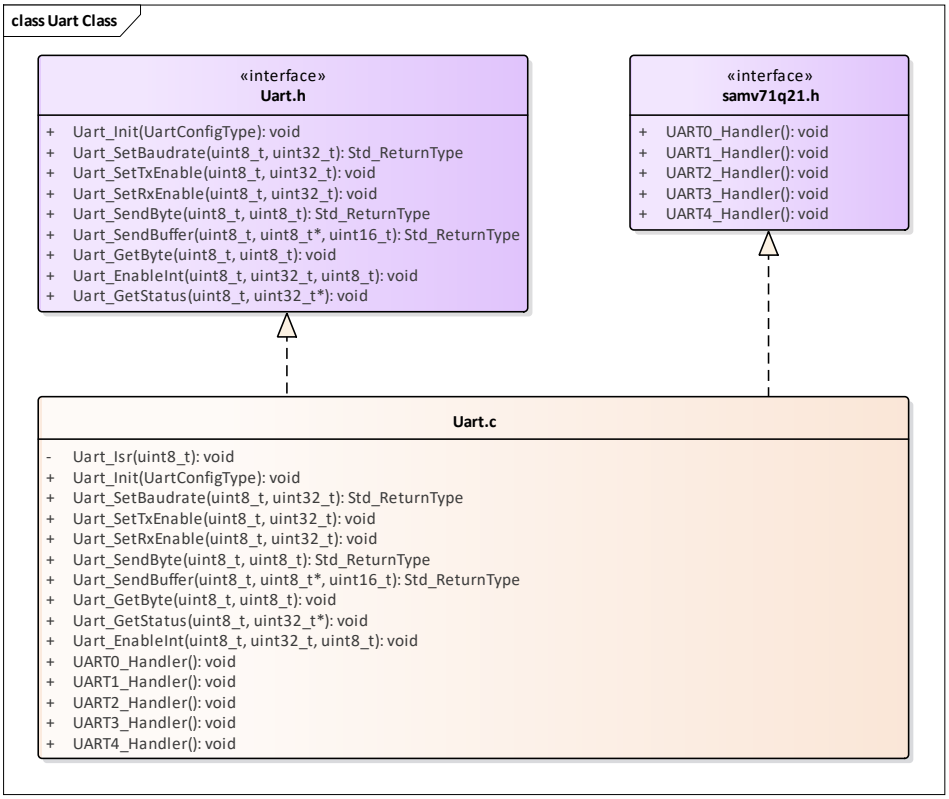
These definitions allow the upper layers to:

- Inform the Uart driver which interrupt to enable or disable through the specific function operations.
- Mask the current status of the Uart module

Refer to Function Definitions chapter for more information of the usage.

3.3. FUNCTION DEFINITIONS

This is a list of functions provided to upper layer modules.



3.3.1. UART_INIT

Service Name	Uart_Init	
Syntax	void Uart_Init (const UartConfigType* Config)	
Sync/Async	Synchronous	
Param (in)	Config	Pointer to Uart static configuration
Param (out)	None	
Return value	None	
Description	Initializes the UART module	

The Uart_Init function shall initialize the UART module. Note that different set of configurations may be provided.

Initialization shall be according to the configuration set pointed by the parameter Config.

3.3.2. UART_SETBAUDRATE

Service Name	Uart_SetBaudrate	
Syntax	Std_ReturnType Uart_SetBaudrate (uint8_t Channel, uint32_t Baudrate)	
Sync/Async	Synchronous	
Param (in)	Channel	UART Channel to be addressed
	Baudrate	Baudrate to configure
Param (out)	None	
Return value	Std_ReturnType	E_OK: Command succesfully executed
		E_NOK: Command could not be executed
Description	Sets the requested baudrate to the addressed UART channel	

The Uart_SetBaudrate function shall support runtime re-configuration of the Uart channel to the specified baudrate parameter.

3.3.3. UART_SETTXENABLE

Service Name	Uart_SetTxEnable	
Syntax	void Uart_SetTxEnable (uint8_t Channel, uint32_t Enable)	
Sync/Async	Synchronous	
Param (in)	Channel	UART Channel to be addressed
	Enable	Enable/Disable information 0: Disable 1: Enable
Param (out)	None	
Return value	void	
Description	Enables or disables the transmitter of the UART module	

The Uart_SetTxEnable function shall support runtime enable/disable of the Uart transmitter specified by the Enable parameter.

3.3.4. UART_SETRXENABLE

Service Name	Uart_SetRxEnable	
Syntax	void Uart_SeRxEnable (uint8_t Channel, uint32_t Enable)	
Sync/Async	Synchronous	
Param (in)	Channel	UART Channel to be addressed
	Enable	Enable/Disable information 0: Disable 1: Enable
Param (out)	None	
Return value	void	
Description	Enables or disables the receiver of the UART module	

The Uart_SetRxEnable function shall support runtime enable/disable of the Uart receiver specified by the Enable parameter.

3.3.5. UART_SENDBYTE

Service Name	Uart_SendByte	
Syntax	Std_ReturnType Uart_SendByte (uint8_t Channel, uint8_t Byte)	
Sync/Async	Asynchronous	
Param (in)	Channel	UART Channel to be addressed
	Byte	Data to be sent over the UART bus
Param (out)	None	
Return value	Std_ReturnType	E_OK: Command succesfully executed
		E_NOK: Command could not be executed
Description	Sends one packet of data through the specified UART module	

The function Uart_SendByte shall send a new byte over the UART bus. If Tx callback notification is configured, the UART module shall notify to the upper layer that the byte transmission is finished.

In order to start a new transmission of data, the upper layer module can:

- Read the status of the Uart channel from the Uart_GetStatus function.
- Be notified that the transmission is finished from the callback notification.
- Poll this function until E_OK is returned to continue the transmit operation.

If the transmitter is busy then E_NOK shall be returned.

If the function was able to write the transmitter register to start a transmission then E_OK shall be returned.

Hint: Internal data status can be implemented to ensure that the driver (Tx interrupt enabled and Tx notification is not configured) and the UART HW module are ready to start a new transmission.

3.3.6. UART_SENDBUFFER

Service Name	Uart_SendBuffer	
Syntax	Std_ReturnType Uart_SendBuffer (uint8_t Channel, uint8_t *Buffer, uint16 Length)	
Sync/Async	Asynchronous	
Param (in)	Channel	UART Channel to be addressed
	Buffer	Pointer to the start address of the buffer to be sent over the UART bus
	Length	Size of the buffer to be sent
Param (out)	None	
Return value	Std_ReturnType	E_OK: Command succesfully executed
		E_NOK: Command could not be executed
Description	Sends a packet of data through the specified UART channel	

The function Uart_SendBuffer shall send a packet of data over the UART bus. The number of data to be sent shall be specified by the Length parameter. If Tx callback notification is configured, the UART module shall notify to the upper layer that the buffer transmission is finished.

In order to start a new transmission of data, the upper layer module can:

- Read the status of the Uart channel from the Uart_GetStatus function.
- Be notified that the transmission is finished from the callback notification.
- Poll this function until E_OK is returned to continue the transmit operation.

If the transmitter is busy then E_NOK shall be returned.

If the function was able to write the transmitter register to start a transmission then E_OK shall be returned.

Hint: Internal data status can be implemented to ensure that the driver (Tx interrupt enabled and Tx notification is not configured) and the UART HW module are ready to start a new transmission.

3.3.7. UART_GETBYTE

Service Name	Uart_GetByte	
Syntax	void Uart_GetByte (uint8_t Channel, uint8_t Byte)	
Sync/Async	Asynchronous	
Param (in)	Channel	UART Channel to be addressed
	Byte	Data received from the UART bus
Param (out)	None	
Return value	void	
Description	Reads and returns a character from the UART module	

The function Uart_GetByte shall return the new byte received from the UART bus. If Rx callback notification is configured, the UART module shall notify to the upper layer that a new data is ready to be read.

In order to get the new byte, the upper layer module shall:

- Read the status of the Uart channel from the Uart_GetStatus function.
- Be notified that the reception is finished from the callback notification.

Hint: Internal data status can be implemented to ensure that the driver has read the data (Rx interrupt enabled and Rx notification is not configured) and the UART HW module have received a new data.

3.3.8. UART_GETSTATUS

Service Name	Uart_GetStatus	
Syntax	void Uart_GetStatus (uint8_t Channel, uint32_t *Status)	
Sync/Async	Synchronous	
Param (in)	Channel	UART Channel to be addressed
Param (out)	Status	Current status of the addressed UART module
Return value	void	
Description	Reads and returns the current status of the addressed UART module	

The Uart_GetStatus function shall return the raw data from the Uart status register.

Note: To support the upper layer to understand the meaning of the raw data, the Uart module provides defined masks specified in the Uart Mask Definitions Chapter.

3.3.9. UART_ENABLEINT

Service Name	Uart_EnableInt	
Syntax	void Uart_EnableInt (uint8_t Channel, uint32_t IntMode, uint8_t Enable)	
Sync/Async	Synchronous	
Param (in)	Channel	UART Channel to be addressed
	IntMode	Interrupt Mode information
	Enable	Enable/Disable information 0: Disable 1: Enable
Param (out)	None	

Return value	void	
Description	Reads and returns the current status of the addressed UART module	

The Uart_Enable function shall enable/disable the UART module interrupts according to the IntMode and Enable parameters.

Note: To support the upper layer to set and read the interrupt mode parameter, the Uart module provides defined masks specified in the Uart Mask Definitions Chapter.

4. DEPENDENCIES TO OTHER MODULES

4.1. FILE STRUCTURE

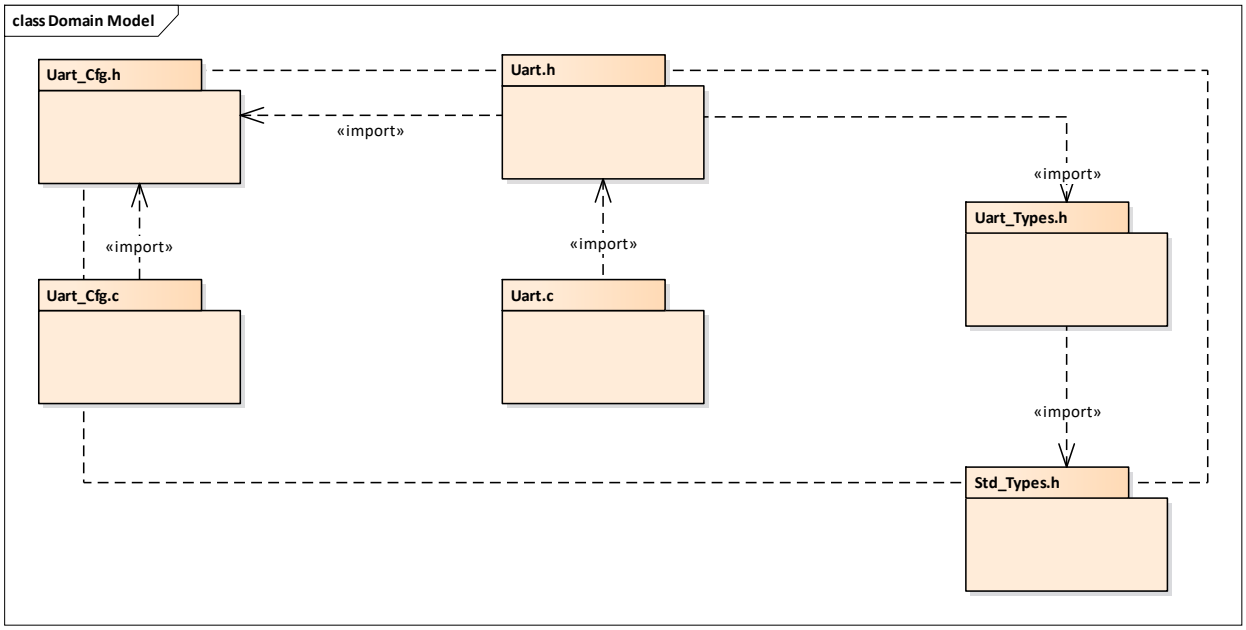
4.1.1. CODE FILE STRUCTURE

The code file structure shall not be defined within this specification completely. It shall be pointed out that the code-file structure shall include the following files named:

- Uart_Cfg.h – for definition configurable parameters, UART configuration types and
- Uart_Cfg.c – for configurable parameters.

4.1.2. HEADER FILE STRUCTURE

The included file structure shall be as follows.



2 HEADER FILE STRUCTURE FOR THE UART DRIVER

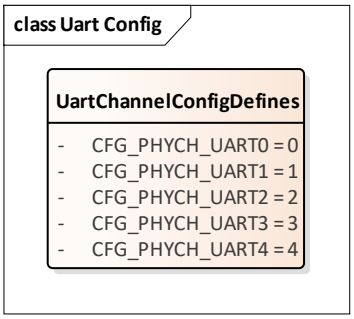
5. UART CONFIGURATION SPECIFICATION

The chapter defines the static configuration parameters and their structure (containers) of the module Uart driver.

The following definitions shall be exported from *Uart_Cfg.h* file.

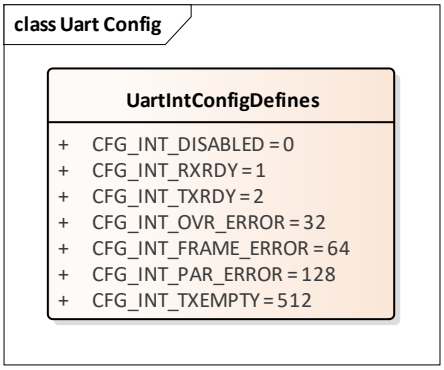
5.1. CONFIGURATION DEFINITIONS

5.1.1. UART CHANNEL CONFIGURATION DEFINITIONS



The above definitions shall serve the user to provide a proper static channel configuration.

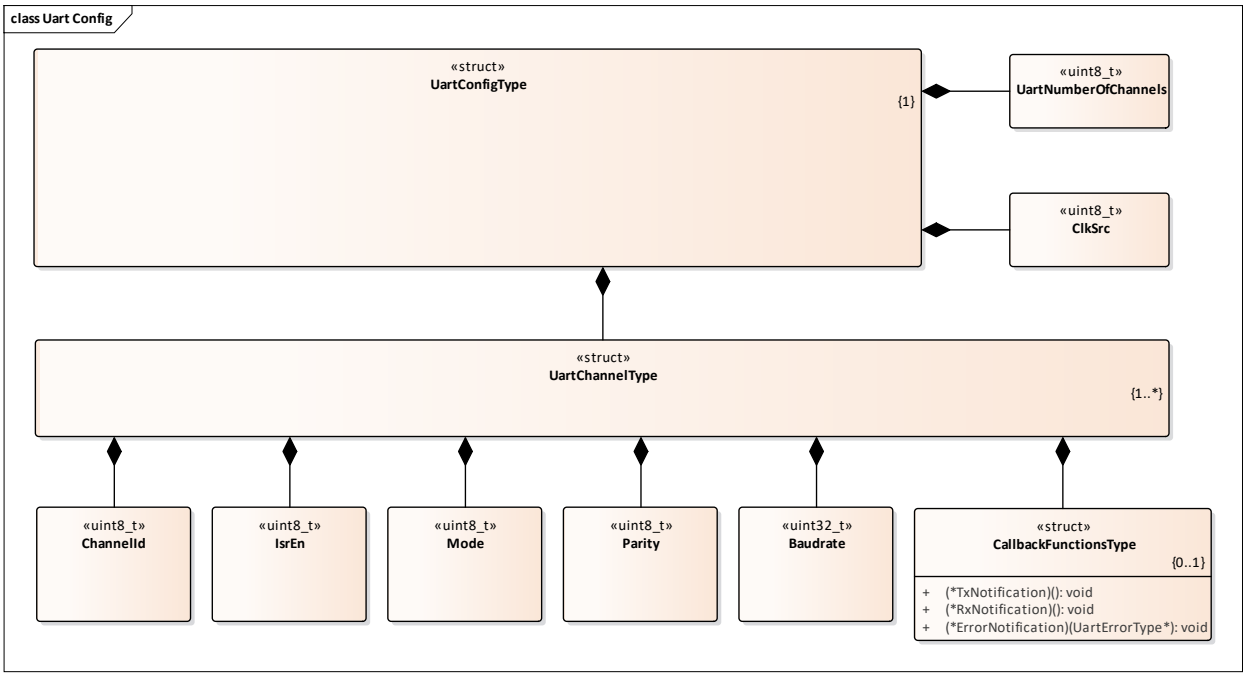
5.1.2. INTERRUPT CONFIGURATION DEFINITIONS



The above definitions shall serve the user to provide a proper static channel configuration.

5.2. CONTAINERS AND CONFIGURATION PARAMETERS

Configuration parameters define the variability of the generic part(s) of an implementation of a module. The main purpose is to provide a configurable module which can be adapted to the environment according to the target hardware and application in use.



5.2.1. UARTCONFIG

Name	*UartConfig*
Type	UartConfigType
Description	Configuration of the UART (Uart driver) module

Included Containers		
Container Name	***Multiplicity***	***Description***
UartChannel	1..*	This container contains the parameters related to each Uart channel

Name	UartNumberOfChannels
Type	uint8_t
Description	Number of channels to be configured
Multiplicity	1
Range	1 .. 255

Name	ClkSrc
Type	uint8_t
Description	Clock Source 0: Peripheral Clock 1: Programmable Clock
Multiplicity	1
Range	0..1

5.2.2. UART CHANNEL

Name	*UartChannel*
Type	UartChannelType

Description	This container contains the configuration parameters of the Uart channel

Included Containers		
Container Name	***Multiplicity***	***Description***
CallbackFunctions	0..1	This container contains the callback notification to upper layers upon transmission, reception and error.

Name	ChannelId
Type	uint8_t
Description	Physical Uart Channel Identifier
Multiplicity	1
Range	1 .. 255

Name	IsrEn
Type	uint32_t
Description	Uart Interrupts enable (Transmission, Reception and Error) Configuration values shall be used as per Interrupt Configuration Definitions chapter. **Note:** Configuration values must be OR'ed
Multiplicity	1
Range	0..4294967295

Name	Mode
Type	uint8_t
Description	Uart Channel Mode 0: Normal 1: Loopback
Multiplicity	1
Range	1 .. 255

Name	Parity
Type	uint8_t
Description	Parity Type 0: Even 1: Odd
Multiplicity	1
Range	1 .. 255

Name	Baudrate
Type	uint32_t
Description	Specifies the baud rate of the Uart channel in bits per second
Multiplicity	1
Range	1..4294967295

5.2.3. CALLBACKFUNCTIONS

Name	*CallbackFunctions*
Type	CallbackFunctionsType
Description	This container contains the configuration parameters of the Uart callback notifications

Not Included Containers

Callback Name	TxNotification	
Syntax	void (*TxNotification) (void)	
Sync/Async	Asynchronous	
Param (in)	None	
Param (out)	None	
Return value	None	
Description	End of transmission notification	

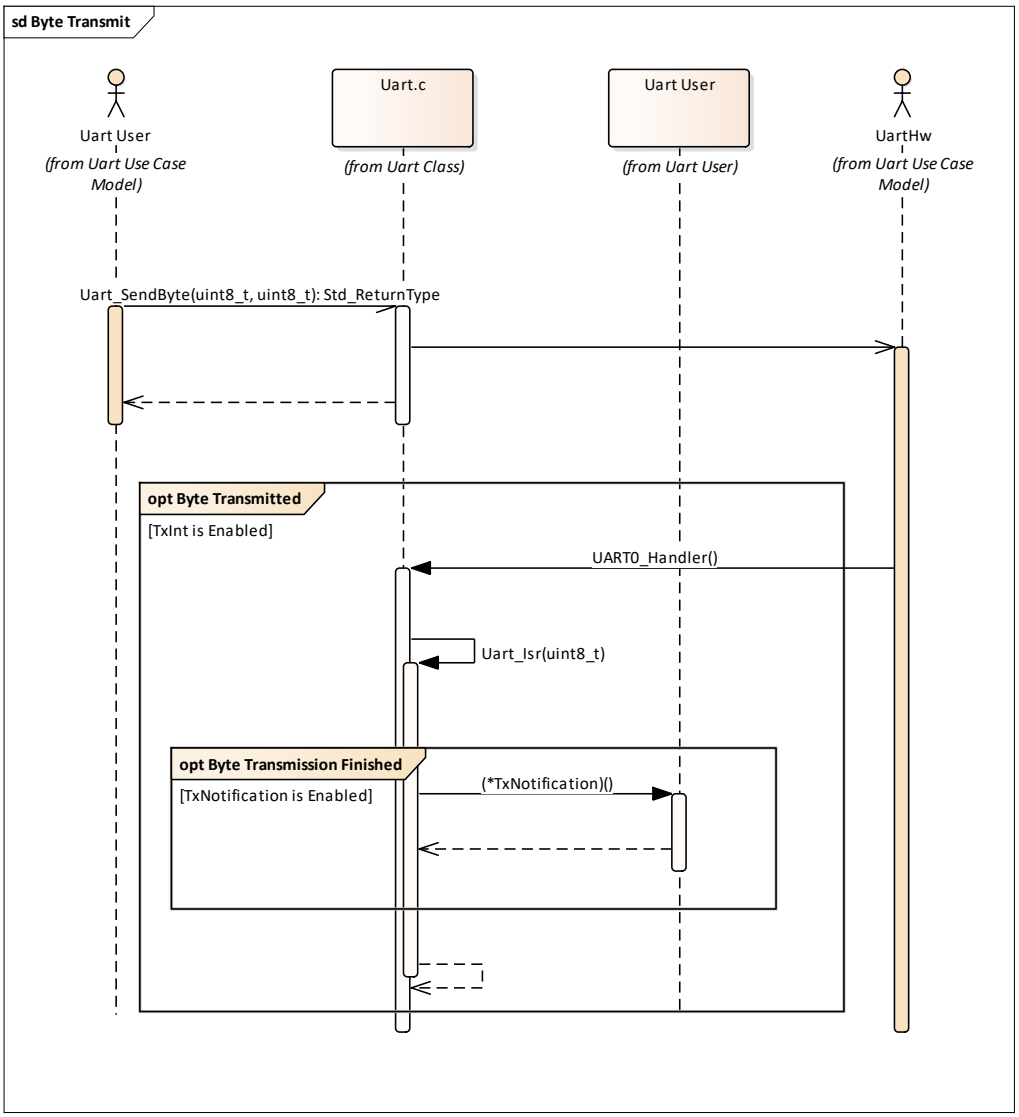
Callback Name	RxNotification	
Syntax	void (*RxNotification) (void)	
Sync/Async	Asynchronous	
Param (in)	None	
Param (out)	None	
Return value	None	
Description	Data reception notification	

Callback Name	ErrorNotification	
Syntax	void (*ErrorNotification) (UartErrorType Error)	
Sync/Async	Asynchronous	
Param (in)	None	
Param (out)	UartErrorType Error	Uart Error during transmission/reception
Return value	None	
Description	Error notification	

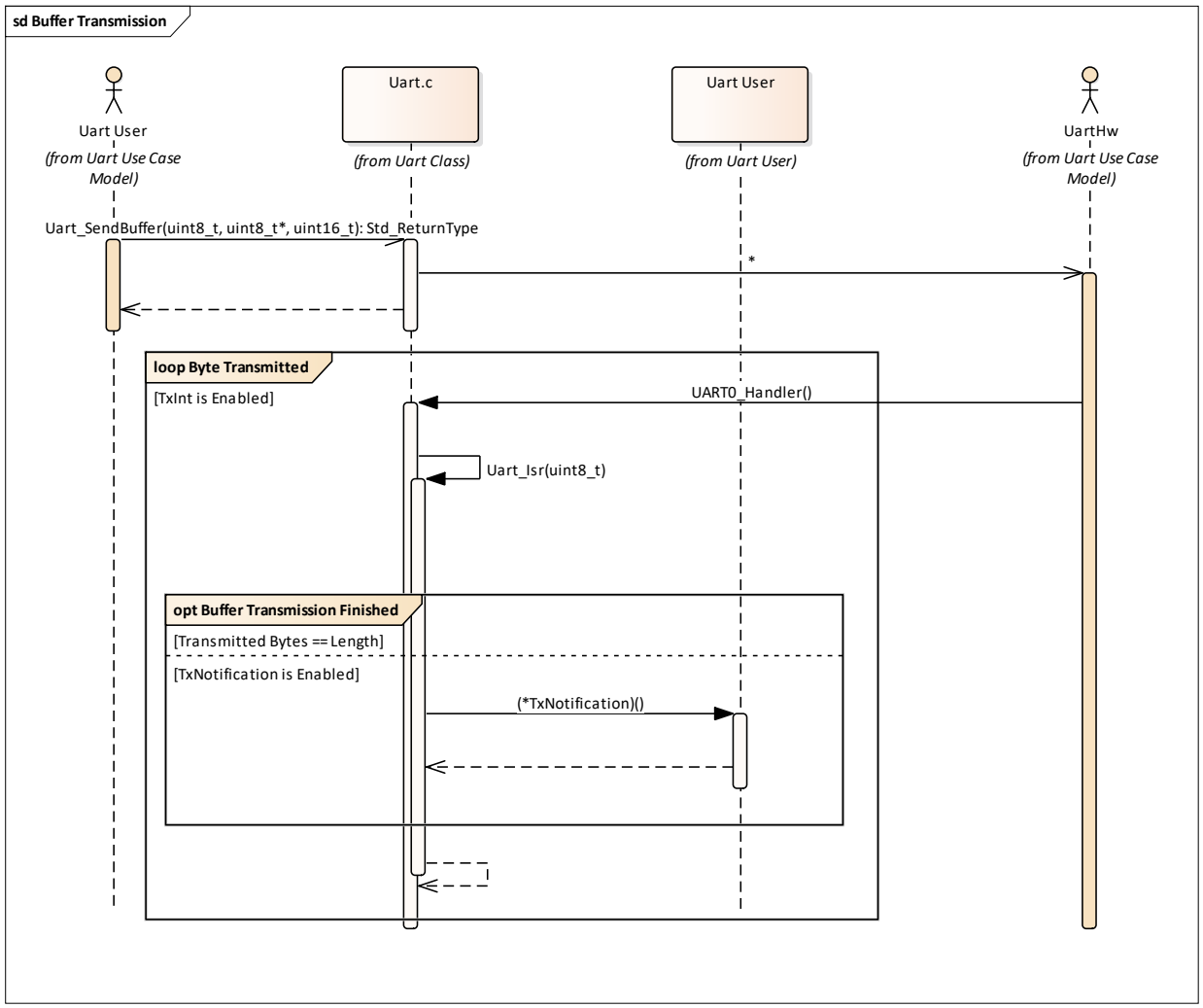
6. UART DYNAMIC DIAGRAMS

6.1. UART TRANSMIT SEQUENCE

6.1.1. BYTE TRANSMIT

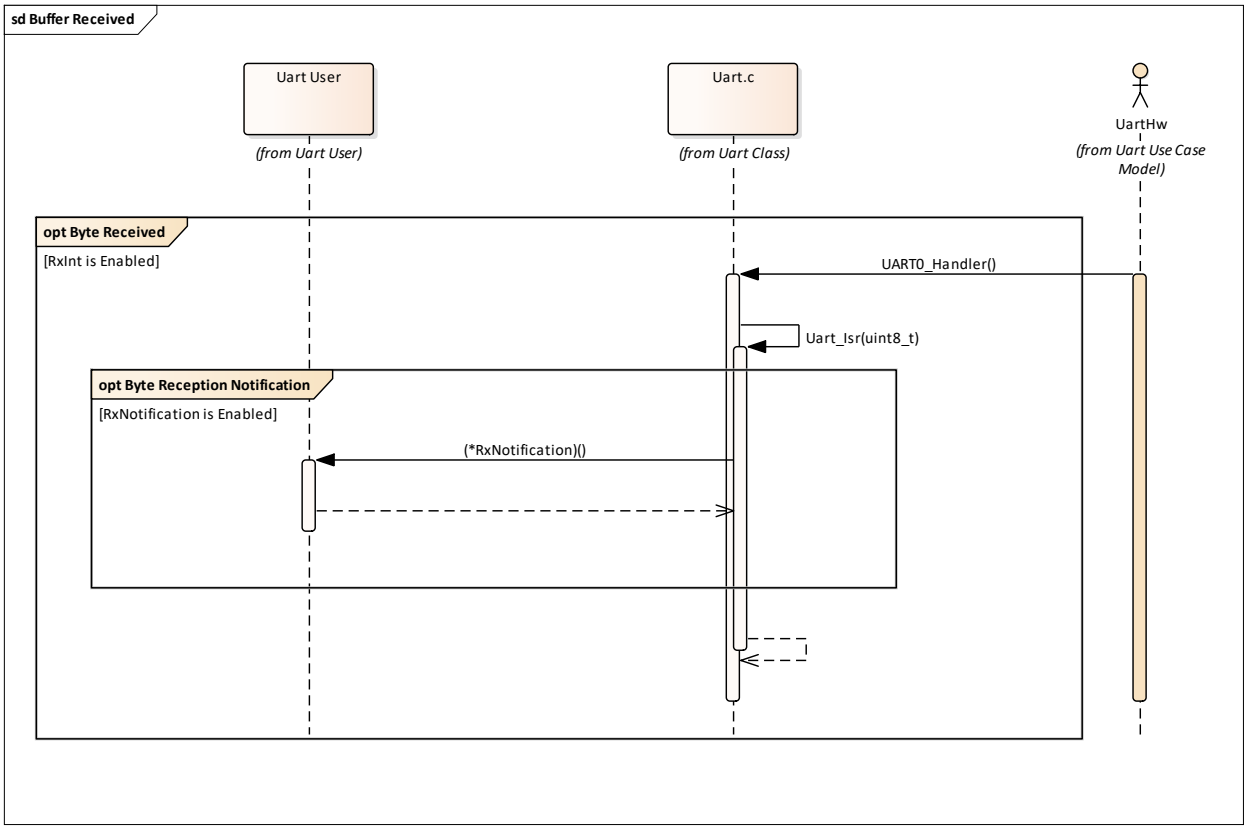


6.1.2. BUFFER TRANSMIT

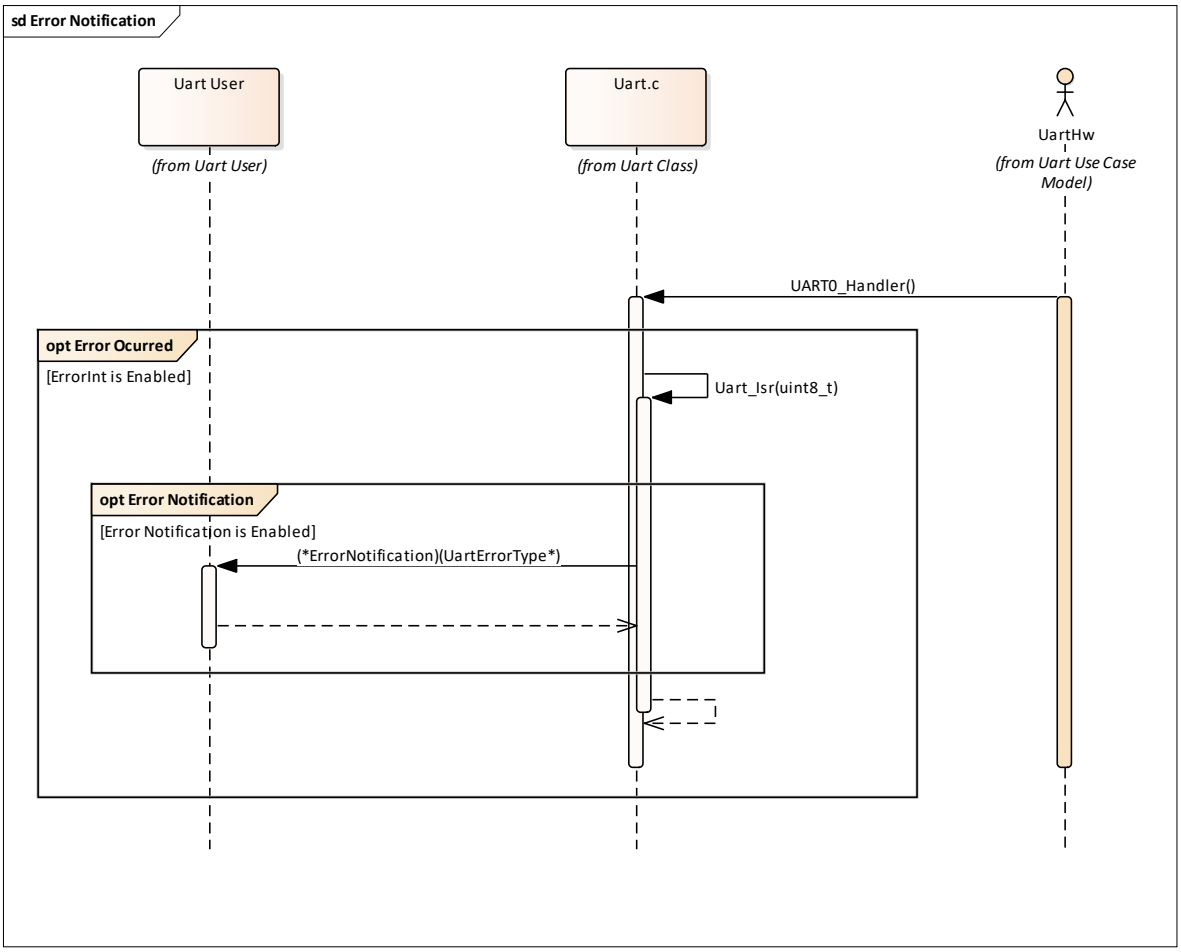


6.2. UART RECEIVE SEQUENCE

6.2.1. BYTE RECEIVE



6.3. UART ERROR SEQUENCE



7. REFERENCES

Document	***Description***
SAM V71Q Datasheet	Atmel-44003-32-bit-Cortex-M7-Microcontroller-SAM-V71Q-SAM-V71N-SAM-V71J_Datasheet